2007

مقدمة في البرمجة بلغة C

أساسيات اللغة

هيكلية البرامج

التراكيب الشرطية

الحلقات التكرارية

المصفوفات

الدوال

محمد سالم البهادلي

المحتويات

بسم الله الرحمن الرحيم

الحمد لله رب العالمين

والصلاة والسلام على سيد الاولين والاخرين ابو القاسم محمد وال بيته الطيبين الطاهرين واصحابة المنتجبين .

الاهداء

الى الملاك الطاهر ...

ذي القلب الكبير والنفس اللوامة المطمئنة ...

والذي تعلق بربه فدعاه خوفاً وطمعاً ...

وما هو الا قبضة من رحمة وعطف وحنان ...

لقد انفصل جسمي عن جسمه ...

لكن روحي لم تزل بروحه متصلة ...

وبهديه وارشاده عرفت جادة الصواب ...

واستبأنت سبيل الرشاد ...

فرضى الله عنه وأرضاه ...

الى من استلهمت منه أسباب البقاء ...

الى أمي ... بعد استئذان ... أبي ... العظيم ...

محمد البهادلي

أسئلكم الدعاء لي ولوالدي

مقدمة:

أصل لغة C ،هي لغة CPL والتي اشتهرت بشدة تعقيدها مما تسبب في ابتعاد المبرمجين عنها،فتم تطويرها وتبسيطها إلى لغة سميت BCPL والتي لم تلقى الكثير من النجاح ولكنها تطورت إلى لغة B ،وبعد فترة قصيرة تم تطوير اللغة B في مختبرات بيل مابين عامي 1969 – 1972 الى لغة برمجة جديدة تضمنت تحديثات وحلول لمعظم المشاكل التي ظهرت للمبرمجين في ذلك الوقت ،سميت هذه اللغة الجديدة بأسم اللغة C (و هذا هو أصل تسمية C). لغة C تمثل اللغة الام للعديد من لغات البرمجة ومنها لغة البرمجة ++C والجافا .

في عام 1978 قام العالمان براين كرنيكان (Brian Kernighan) و دنس رتشي (Dennis Ritchie) بنشر أول كتاب يتضمن اللغة C بصورة مفصله تحت عنوان (The C Programming Language)، حيث أصبح هذا الكتاب المرجع الاساسي في اللغة C حتى يومنا هذا.واللغة C قد صممت في الأصل لتستعمل في تطوير و العمل تحت نظام التشغيل يونكس، ثم لقيت انتشارا واسعا منذ ذلك الحين و تواصل إلى اليوم ويظهر ذلك من خلال شعبيتها لدى أغلب مبرمجي الحاسوب و من خلال الاستعمالات العديدة والمتنوعة.

الفصل الأول: أساسيات C (Fundamentals of C)

تُستخدم في لغة C كما هو الحال في أي لغة برمجة أخرى مجموعة من الأساسيات التي يجب التعرف عليها عند دراسة أي لغة برمجة، و هذه الأساسيات هي :

(1.1) الرموز(Characters)

وتتألف من :-

a) الأرقام(*digits*) وهي *9,8,7,6,5,4,3,2,1,0*

b) الحروف الهجائية(*letters*) وهي الحروف الكبيرة *A,B,C,…,X,Y,Z* والحروف الصغيرة *a,b,c,…,x,y,z*

c) الرموز الخاصة(*special characters*) وهي

+	-	*	/	.	)	(	:	$	#	;	,	=	<	>	%	}	{	!	"	'

(1.2) الكلمات المحجوزة (*Reserved words*)

وهي كلمات موجودة في المكتبة القياسية للغة C تستعمل أسماء، ولها وظيفة معينة،ولايمكن استعمالها لغير ما وظفت به لان ذلك سوف يحدث بسببها أرباك للمترجم (Compiler). ومن هذه الكلمات هي :

auto	break	case	catch	char	class	const	continue	default	delete
do	double	else	enum	extern	false	float	for	goto	if
int	long	return	short	signed	sizeof	struct	switch	true	typedef
typename	union	unsigned	using	void	while	and	or		

(1.3) المعّرفات (Identifiers)

المعّرف هو ذلك الاسم الذي يخزن به قيم مثل الثابت أو المتغير أو الدالة ، ومن شروط المعّرف هي :

a. أن يتكون من حرف أو حروف أو حروف وأرقام.
b. خالي من أي رمز خاص ماعدا الرمز(_) under score .
c. أن يبدأ دائماً بحرف أو الرمز (_).
d. يسمح باستخدام الحروف الصغيرة والحروف الكبيرة .
e. أن يكون له الطول المناسب و واضحاً وذا معنى ومدلول .

مثال: معّرفات مقبولة x12, student_name, degree, AVARAGE, ToTaL, sum

مثال:- معّرفات غير مقبولة
يبدأ برقم 7UP
يبدأ بالرمز الخاص $ .... $cost
يمثل كلمة محجوزة .... return
مستخدم فراغاً................. LG NAME

(1.4) الأعداد (Numbers)

الأعداد في لغة C هي :

a. **الاعداد الصحيحة :-** وهي الاعداد الخالية من الفاصلة العشرية، مثال 927- 5476 0 56- 28

b. **الاعداد الحقيقية :-** وهي الاعداد التي تتضمن فاصلة عشرية ، مثال 326.5877- 0.0 31.67 0.67- 16.315

c. **الاعداد الحقيقية ذات الدقة المضاعفة :-** وهي الاعداد الحقيقية الممثلة بشكل قوة آسية باستخدام الحرف (e) أو الحرف (E) حيث يدل الحرف e أو E على القوة. مثال 0.22E9 99.432E-8 1.23e2 72e-65 12.3e-5

(1.5) السلاسل والحروف (Strings and Characters)

السلسلة:- وهي مجموعة من الحروف أو الأرقام أو الرموز الخاصة، بشرط أن تكون موضوعة بين علامتي التنصيص المزدوجة(" ").

مثال:- "56&4787" , "NAME:" , "Go to room" , "This is a sentence."

الحرف:- وهو حرف أو رمز أو رقم موضوع بين علامتي التنصيص المفردة (' ').

مثال:- '7' , '*' , 'a' , 'A' , '+'

(1.6) المتغيرات (Variables)

وهي أسماء رمزية يخصص لها أماكن تخزين في ذاكرة الحاسب، والتي تتغير قيمتها من قيمة الى أخرى،ويمكن الرجوع إليها عن طريق هذه الأسماء أثناء تنفيذ البرنامج.

في لغة C يجب *أن يعلن عن المتغيرات* مسبقاً(أي قبل تخصيص قيمة لها)، *ولا فلن يتعرف لها* مترجم اللغة(Complier).
ولغرض الاعلان عن المتغيرات نستخدم الصيغة الآتية :

```
Type Variable_Name;
```

حيث **Variable_Name:** يمثل اسم المتغير، والذي يخضع الى الشروط التالية :

a. أن يتكون من حرف أو حروف أو حروف وأرقام .
b. خالي من أي رمز خاص ماعدا الرمز(_) under score .
c. أن يبدأ دائماً بحرف أو الرمز (_).
d. يسمح باستخدام الحروف الصغيرة والحروف الكبيرة مع الانتباه الى أن لغة C تميز بين الاحرف الكبيرة والصغيرة .
e. أن يكون له الطول المناسب ،حيث أن اقصى طول يمكن أختيارة لأسم متغير هو 32 حرف أو رقم.
f. واضحاً وذا معنى ومدلول (أي يفضل أختيار اسم المتغير تبعاً للبيانات التي يحملها ، فمثلاً نختار أسم المتغير age للعمر أو degree للدرجة أو average للمعدل،هكذا...).

Type: يمثل نوع المتغير، أي نوع القيمة التي يحملها ذلك المتغير، ويمكن ان يكون احد الأنواع الموجودة في الجدول الأتي:

النوع	الوصف	الحجم	المدى
short int short	Short Integer	2bytes	-32768 to 32767
unsigned short int unsigned short	Short Integer	2bytes	0 to 65535
int	Integer	2bytes Or 4bytes	يعتمد على النظام -32768 to 32767 or -2147483648 to 2147483647
unsigned int	Integer	4bytes	0 to 4294967295
long int long	Long integer	4bytes	-2147483648 to 2147483647
unsigned long int unsigned long	Long integer	4bytes	0 to 4294967295
bool	Boolean value: true or false	1byte	true or false
float	Floating point number	4bytes	3.4e- 38 to 3.4e+38 (7 digits)
double	Double precision floating point number	8bytes	1.7e- 308 to 1.7e+308 (15 digits)
long double	Long double precision floating point number	10bytes	1.2e- 4932 to 1.2e+4932 (19 digits)
char	Character	1byte	-128 to 127
unsigned char	Character	1byte	0 to 255

مثال (1.6.1) :

Example 1	Example 2	Example 3
int x ; float a ;	float mynumber ; double cost ;	char m ; short int n ;

ملاحظة(1.6.2):- يمكن الاعلان عن أكثر من متغير لنفس النوع بسطر واحد، وكما يلي :

Type Variable_Name1, Varible_Name2, Variable_Name3,……;

Example
int a,b ; float x,y,z ;

ملاحظة(1.6.3):- من الممكن أعطاء قيمة للمتغير أثناء الاعلان عن ذلك المتغير،وكما يلي :

Type Variable_Name=Varible_Value ;

Example 1	Example 2	Example 3
int x = 168 ; float a=b= -78.2 ;	float mynumber = 5.78 ; float sum= 0.0 ;	char m = ' T ';

ملاحظة(1.6.4):- أنشاء الله سوف نتطرق بصورة مفصلة لعمليات أدخال القيم للمتغيرات في الفصل الثاني في موضوع الادخال والاخراج.

(1.7) الثوابت (Constants)

تستخدم الثوابت في لغة C حيث يدل الثابت على قيمة لا يمكن تغيير ها اثناء البرنامج .والصيغة العامة للثابت هي:

const Type constant_name=constant_value ;

حيث أن Type : يمثل نوع قيمة الثابت.
constant_name: يمثل اسم الثابت.
constant_value : تمثل قيمة الثابت.

Examples
const int pathwidth= 100 ; const char tabulator= '\t' ; const char ch= "C Good Lang."; const double PI= 3.14159265 ;

(1.8) المؤثرات(Operators)

يوجد في لغة C عدد من المؤثرات ،وهي كالاتي:-

a) **المؤثرات الحسابية(Arithmetic Operators):** وهي

المؤثر	معناه	مثال
+	addition	a+b
-	subtraction	a-b
*	multiplication	a*b
/	division	a/b
%	modulo	a%b

ملاحظة(1.8.1) :- يجب أن ناخذ بعين الاعتبار بالنسبة لمؤثر باقي القسمة (%) يجب أن تكون عناصر البيانات المستعملة قيم صحيحة والا فان النتيجة تكون خاطئة.

مثال :- اذا كان a=11 و b=3 فان a%b=2 ،

اما اذا كان a أو b أو كلاهما عدداً حقيقياً، فان عملية باقي القسمة لا يمكن تطبيقها ،لان النتيجة تكون خاطئة .

b) المؤثرات العلائقية(Relational Operators): وهي ست مؤثرات تستخدم على أي زوج من العناصر ويكون ناتجها اما صحيحاً True (أي قيمة ما عدا 0) أو خطئاً False(قيمة 0) وهي كما يلي:

المؤثر	معناه	مثال	النتيجة
= =	Equal to	7= =5	False (0)
!=	Not equal to	3!=2	True (1)
<	Less than	4>9	True (1)
< =	Less than or equal to	7< =6	False (0)
>	Greater than	-2>4	False (0)
> =	Greater than or equal to	12> =10	True (1)

c) المؤثرات المنطقية(Logical operators): وهي مؤثرات ينتج عنها اما قيمة صحيح TRUE (العدد 1) أو قيمة خطأ FALSE (أي قيمة ما عدا 1) .

المؤثر	معناه
&&	And
||	Or
!	negating or anti-thesis

a	b	a && b	a||b	!a
True	True	True	True	False
True	False	False	True	
False	True	False	True	True
False	False	False	False	

d) المؤثرات الخاصة بالبت(Bitwise Operators): وهي

المؤثر	المكافى الى	معناه
&	AND	Bitwise AND
|	OR	Bitwise Inclusive OR
^	XOR	Bitwise Exclusive OR
~	NOT	Unary complement (bit inversion)
<<	SHL	Shift Left
>>	SHR	Shift Right

e) المؤثرات المركبة(Compound Operators): هناك ميزة في لغة C وهي استخدام المؤثرات الحسابية والمؤثرات الخاصة بالبت مع اشارة التخصيص (=) تحت اسم المؤثرات المركبة، وهي طريقة مختصرة لجلة التخصيص . والمؤثرات المركبة هي : += , -= , *= , /= , %= , &= , <<= , >> .

فمثلاً التعبير x=x+9 تعني اضف القيمة 9 للمتغير القديم x الموجود في الطرف الايمن، ثم خصص هذه القيمة الجديدة للمتغير الجديد الموجود في الطرف الايسر وهو x .

وعلى هذا يمكن استخدام التعبير السابق بطريقة المؤثر المركب =+ وكما يلي : x+=9 .

جملة التخصيص	جملة التخصيص باستخدام المؤثر المركب
value = value + increase;	value += increase;
a = a - 5;	a -= 5;
a = a / b;	a /= b;
price = price * (units + 1);	price *= units + 1;

f) **مؤثرات الزيادة والنقصان(Increment Decrement Operators)** هناك ميزة في لغة C قد لا نجدها في بعض لغات البرمجة الاخرى وهي، مؤثر الزيادة(++) ومؤثر النقصان(--) حيث يمكن استعمالهما مع المتغيرات فقط.

مؤثر الزيادة(++): التعبير a++ يعني استخدم القيمة الحالية للمتغير a في حساب التخصيص ، ثم أضف القيمة 1 الى المتغير a . اما التعبير a++ يعني أضف القيمة 1 الى المتغير a، ثم استخدم القيمة الجديدة للمتغير a في حساب التخصيص.

مؤثر النقصان(--): التعبير a-- يعني استخدم القيمة الحالية للمتغير a في حساب التخصيص ، ثم أنقص القيمة 1 من المتغير a . اما التعبير a-- يعني أنقص القيمة 1 من المتغير a، ثم استخدم القيمة الجديدة للمتغير a في حساب التخصيص.

Example 1	Example 2	Example 3	Example 4
A=3; B=++A; /* B contains 4 A contains 4 */	A=3; B=A++; /* B contains 3 A contains 4 */	B=C=3; A=(++B+C++); /* B contains 4,C contains 3 A contains 7 */	B=C=3; A=(B--+C--); /*B contains 3,C contains 3 A contains 6 */

g) **مؤثر الفاصلة (,) (The Comma Operator)** : أذا كان لدينا أكثر من تعبير مفصولة عن بعضها بفاصلة فأن القيمة النهائية تحسب من اليسار الى اليمين، ونوعها هو نوع التعبير بالطرف الايمن.

الصيغة العامة **expression_1, expression_1, … , expression_1 ;**

مثال (1.8.2):-

```
a=2;
b= (a+=4, 12/a);
```

نلاحظ أن التنفيذ في المثال اعلاه يتم كالاتي : تعطى القيمة 2 للمتغير a، ثم يُنفذ مؤثر الفاصلة حيث يُنفذ التعبير الأول a+=4 والذي تنتج عنة القيمة 6 والتي تخصص للمتغير a بعدها يُنفذ التعبير الثاني 12/a والذي تنتج عنة القيمة 3 والتي تخصص الى المتغير b .

(1.9) أولوية تنفيذ المؤثرات (Operators Precedence)

الاولوية في التفيذ تتضح عند وجود تعبير ما به عدد من المؤثرات المختلفة، فاذا اردنا أن ياخذ التعبير مساراً محدداً لتقييمه يجب استخدام الاقواس وذلك حسب ما يقتضيه التعبير،اما الاولوية في التنفيذ فتتم حسب الاتي :

<table>
<tr><th>التسلسل</th><th>المؤثر</th><th>الوصف</th><th>التنفيذ</th></tr>
<tr><td>1</td><td>++ -- ~ !</td><td>unary (prefix)</td><td>Right-to-left</td></tr>
<tr><td>2</td><td>* / %</td><td>multiplicative</td><td>Left-to-right</td></tr>
<tr><td>3</td><td>+ -</td><td>additive</td><td>Left-to-right</td></tr>
<tr><td>4</td><td><< >></td><td>shift</td><td>Left-to-right</td></tr>
<tr><td>5</td><td>< <= > >=</td><td>relational</td><td>Left-to-right</td></tr>
<tr><td>6</td><td>!= ==</td><td>equality</td><td>Left-to-right</td></tr>
<tr><td>7</td><td>&</td><td>bitwise AND</td><td>Left-to-right</td></tr>
<tr><td>8</td><td>^</td><td>bitwise XOR</td><td>Left-to-right</td></tr>
<tr><td>9</td><td>|</td><td>bitwise OR</td><td>Left-to-right</td></tr>
<tr><td>10</td><td>&&</td><td>logical AND</td><td>Left-to-right</td></tr>
<tr><td>11</td><td>||</td><td>logical OR</td><td>Left-to-right</td></tr>
<tr><td>13</td><td>= *= /= %= += -= >>= <<= &= ^= |=</td><td>assignment</td><td>Right-to-left</td></tr>
<tr><td>14</td><td>,</td><td>comma</td><td>Left-to-right</td></tr>
</table>

الفصل الثاني: الهيكلية للبرامج في C (Skeletal of Programs in C)

لغرض التعرف على شكل البرنامج في لغة C ، يجب النظر الى مجموعة من البرامج البسيطة والتي تبين الهيكل العام لاي برنامج باللغة C.

Program 1	التنفيذ
(see code below)	C is Good Language

```
/* my first program in C */
#include <stdio.h>
main ( )
{
 printf ( "C is Good Language" ) ;
}
```

Program 2	التنفيذ
(see code below)	5 – 2 = 3

```
#include <stdio.h>
main ( )
{
 int a, b ;
 float result ;
 a = 5 ;
 b = 2 ;
 result = a – b ;
 printf ( "%d - %d = %f" ,a ,b ,result) ;
}
```

 Email : m.albahadeli@yahoo.com

بعد الاطلاع على المثالين اعلاه،يمكننا الان شرح هيكلية البرنامج بلغة C ،حيث أن هذه الهيكلية تتألف من :

(2.1) التعليقات (Comments)

وهي جمل ايضاحية تستخدم في البرنامج فقط لتسهيل توثيق واعادة قراءة البرنامج أو تعديله من طرف المبرمج أو الاخرين .وهذه التعليقات يمكن أن تكتب في أي مكان من البرنامج، ومن الممكن أن لا تكون موجودة في البرنامج،ولا يكون لها أي تاثير على البرنامج، لان مترجم اللغة يتجاهلُها اثناء التنفيذ .

الصيغة العامة للتعليق : **/* Comment Statement */**

عند استخدام هذه الصيغة يجب الانتباه الى مايلي: اذا كان لدينا أكثر من سطر لجملة التعليق فيجب وضع الرمز(/*) في بداية السطر الاول لجملة التعليق ووضع الرمز(*/) في نهاية السطر الاخير لجملة التعليق .

__مثال__ (2.1.1) :-

1. /* first program in C Language */
2. /* This Program calculate the Average

   of n numbers using C Language */

(2.2) التوجيهات (Directives)

التوجيه هو عملية ادراج ملف عنوان رأسي(Header File) ضمن البرنامج، حيث أن هذا الملف موجود ضمن ملفات المكتبة القياسية للغة C والذي يحتوي على بعض الايعازات المستخدمة ضمن البرنامج والمطلوب التعرف عليها وتنفيذها من قبل مترجم اللغة.من هذه الملفات (stdio) والذي يحتوي على عبارات الادخال والاخراج والعمليات الحسابية، وهذه الملف يجب ادراجه ضمن جميع برامج اللغة C، الملف الاخر هو (math) والذي يحتوي على الدوال الرياضية(الدوال المثلثية، الدوال اللوغارتيمية، الدوال الاسية،.....). وكذلك يوجد العديد من الملفات والتي سوف نتطرق لها من خلال البرامج اللاحقه.

اما الصيغة العامة للتوجيه هي :- **#include< File_Name.h >**

حيث

File_Name : يمثل اسم الملف.

Example: - **(1)** #include< stdio.h >
(2) #include< math.h >

(2.3) الدالة الرئيسية main( ) (Main Function)

وهي الدالة التي يبدأ بها البرنامج،وتكون موجودة في جميع البرامج بلغة C ،

الصيغة العامة
main( ) **{** **Program Body ;** **}**

حيث Program body يمثل جملة أو مجموعة من الجمل المطلوب تنفيذها.

(2.4) إدخال وإخراج البيانات (Input/Output Data)

***(1)* الادخال:-** (راجع موضوع المتغيرات) نلاحظ أنه تم أستخدام مؤثر التخصيص (=) لتخصيص قيم لمتغيرات، وهذا لا يسمح بتغير تلك القيم الا بتغير جملة التخصيص حيث تكون ثابتة اثناء تنفيذ البرنامج . لذلك يُفضل في معظم البرامج استخدام دالة الادخال scanf( ) والموجودة

ضمن ملف العنوان stdio.h، وهذه الدالة تأخذ المعطيات من لوحة المفاتيح وتخصصها لاسماء متغيرات، حيث يمكن استخدامها في البرنامج فيما بعد .

الصيغة العامة لدالة الادخال هي: **scanf ("Format" , &Variable_Name) ;**

حيث Format : تعني التوصيفات المختلفة التي تحتوية دالة الادخال .الجدول التالي يوضح التوصيفات الخاصة بأدخال البيانات.

المؤثر	المعنى
%d	يستخدم للعدد الصحيح
%f	يستخدم للعدد الحقيقي
%c	يستخدم للحرف
%s	يستخدم للسلسلة
%u	يستخدم للعدد الصحيح بدون اشارة
%o	يستخدم للعدد الصحيح بالنظام الثماني
%x	يستخدم للعدد الصحيح بالنظام السادس عشر
%e	يستخدم للعدد الصحيح بالصورة الاسية

مثال(2.4.1) :- لاحظ الجملة الاتية : scanf("%f" , &x) ;

عندما يصل التنفيذ الى هذه الجملة،يتوقف منتظراً أدخال عدد حقيقي للمتغير x عن طريق وسيلة الادخال لوحة المفاتيح،ثم تخزن تلك القيمة في عنوان المتغير x المخصص له في الذاكرة .

ملاحظة(2.4.2):- اذا كان لدينا اكثر من متغير، فأنة بالامكان استخدام الدالة scanf ( ) لادخال القيم لهذه المتغيرات.وكما يلي

scanf ("Format" , &Variable_1,&Variable_2,&Variable_3, … ,& Variable_n);

مثال(2.4.3) :- لاحظ الجملة الاتية : scanf("%d%f%f%c" , &a,&b,&c,&op) ;

(2) الاخراج:- تستخدم الدالة printf( ) لاخراج البيانات على وحدة الاخراج القياسية(الشاشة) ، وهذه الدالة موجودة ضمن ملف العنوان stdio.h .

الصيغة العامة لهذة الدالة هي :

printf(" Format ",arg_1,arg_2,..., arg_n) ;

حيث
Format : تعني التوصيف الازم للطباعة،والتي تتضمن عبارات الاخراج المناسبة والمنسقة للبيانات،وكذلك تتضمن التوصيفات الخاصة والموجودة ضمن الجدول الخاص بتوصيفات دالة الادخال scanf( ) .

ملاحظة(2.4.3):- البرنامج الجيد هو البرنامج الذي تكون مخرجاته منسقة وذات توصيف جيد للبيانات.
أما arg_1, arg_2, arg_3, ….. ,arg_n : فهي عناصر البيانات ، وهي اختصار لكلمة الادلة(Arguments) ويمكن أن تكون هذه الادلة ثوابت عددية أو متغيرات من النوع (الصحيح – الحقيقي –الحرفي) المطلوب طباعتها على الشاشة، وعلى أن تفصل الادلة عن بعضها بواسطة فواصل .

ملاحظة(2.4.4):- هناك بعض الرموز الخاصة، يطلق عليها أحياناً (رموز الهروب)، و الممكن استخدامها مع دالة الاخراج printf( ) للتحكم في المخرجات أو الطباعة على الشاشة، وهذه الرموز يمكن أن توضع ضمن التوصيفات(Format) أو بصورة منفردة وفي حالة كتابتها بصورة منفردة فيجب وضعها داخل علامة التنصيص المزدوجة(" ") و هي كالاتي :

معناه	الرمز	معناه	الرمز
الانتقال الى صفحة جديدة	\f	الانتقال الى سطر جديد	\n
استخدام الجرس	\a	البدء من أول السطر	\r
لطباعة الحرف	\\	التراجع مسافة الى الخلف	\b
لطباعة علامة التنصيص المفردة	\'	التقدم مسافة معينة قبل الطباعة	\t
		لطباعة علامة التنصيص المزدوجة	\"

مثال(2.4.5) :-

Program	التنفيذ
#include <stdio.h> main ( ) { printf ("Hello World! ") ; printf ("I'm a C program") ; }	Hello World! I'm a C program

نلاحظ من خلال هذا المثال أن التفيذ قد أظهر الطباعة للعبارتين بسطر واحد ،وهذا تنسيق غير جيد للطباعة اذ اننا نريد طباعة كل عبارة بسطر واحد مستقل عن السطر الاخر، لذلك سوف نستخدم الرموز **\n** لهذا الغرض .

Program	التنفيذ
#include <stdio.h> main ( ) { printf ("Hello World!\n ") ; printf ("I'm a C program") ; }	Hello World! I'm a C program

مثال(2.4.6):- أكتب برنامجاً لحساب وطباعة مساحة ومحيط دائرة نصف قطرها r =5.2 .

Program

```
/* calculate area and circumference */
#include <stdio.h>
main ()
{
  const double PI=3.1415926 ;
  float r=5.2;
  double area ,circumference ;
  area = PI * r * r ;
  circumference = 2 * PI * r ;
  printf("The Area of circle =%f" ,area ) ;
  printf("\n The Circumference of circle =%f",circumference) ;
}
```

التنفيذ

```
The Area of circle =84.948663
The Circumference of circle =32.672563
```

مثال(2.4.7):- المطلوب كتابة برنامج لادخال الطول والعرض لمستطيل ما ،ثم حساب وطباعة محيط ومساحة هذا المستطيل .

Program	التنفيذ
#include <stdio.h> main() { int length, width ; int perimeter, area ; printf("Length =") ; scanf("%d",&length) ; printf("\n Width =") ; scanf("%d",&width) ; perimeter = 2*(length+width) ; area = length*width ; printf("Perimeter is %d " ,perimeter) ; printf("\n Area is %d", area) ; }	Length= 20 Width= 15 Perimeter is 70 Area is 300

مثال(2.4.8):- أكتب برنامجاً لقراءة درجات طالب في أربعة مواد (95 69 82 77)، وحساب وطباعة معدل الطالب .

Program	التنفيذ
#include <stdio.h> main() { int d1, d2, d3, d4, sum = 0 ; float average ; printf("Enter Four Degree : ") ; scanf("%d%d%d%d",&d1,&d2,&d3,&d4) ; sum =(d1+d2+d3+d4) ; average = sum/4 ; printf("\n") ; printf("The Average of Student is %f ", average) ; }	Enter Four Degrees : 77 82 69 95 The Average of Student is 80.75

ملاحظة(2.4.9):- المبرمج الجيد، دائما يبحث عن كتابة برنامجة بأقل الخطوات ، لذلك يمكن أعادة المثال اعلاه بالبرنامج الاتي :

Program	التنفيذ
#include <stdio.h> main() { int d1, d2, d3, d4 ; cout << "Enter Four Degree : " ; scanf("%d%d%d%d",&d1,&d2,&d3,&d4) ; printf("\n The Average of Student is %f " << (d1+d2+d3+d4)/4 ; }	Enter Four Degrees : 77 82 69 95 The Average of Student is 80.75

تمارين (الفصل الثاني)

تمرين 1 :- أطبع أسمك كاملاً في السطر الاول من الجهة اليسرى،وأطبع عنوانك في وسط السطر الثالث،ثم أطبع الجنسية في السطر من الجهة اليمنى من الشاشة.

تمرين 2 :- أكتب برنامجاً يقرأ درجة الحرارة بالدرجة الفهرنهايتية، ثم يطبعها بالدرجة المئوية مستخدماً المعادلة $C = \frac{5}{9}(F - 32)$

تمرين 3 :- أكتب برنامجاً لادخال قيم حقيقية للمتغيرات x , y , z , w ، وحساب وطباعة كل مما ياتي :

$A = \frac{5+x}{z} + \frac{y}{2.7w}$ 2) $B = \frac{4.5(x+2.3y)^2}{z+w}$

تمرين 4 :- أكتب برنامجاً لقراءة العجلة الثابتة A والزمن T ، ثم أحسب *وأطبع المسافة* D والسرعة النهائية V ، *اذا علمت أن* $D=0.5AT^2$ وأن السرعة النهائية هي $V=AT$.

تمرين 5 :- تتبع واستنتج مخرجات البرنامج التالي :

Program

```
#include <stdio.h>
main ( )
{
  short a,b,c ;
  short d,e ;
  d=e=30 ;
  a=4 ;
  b=--a+1 ;
  c=++a + b++ ;
  printf( "A=%d \t B=%d \t C=%d\n",a,b,c );
  c+=--a + --b ;
  a=b=c-(a*b) ;
  d/=a+b ;
  e=e/a +b ;
  printf( "\n D=%d \t E=%f" ,d,e) ;
}
```

تمرين 6 :- أكتب برنامجاً لادخال الجنس(Sex) وفصيلة الدم لزوجين، وأظهار النتائج كالاتي

SEX	BLOOD SPECIES
M	A+
F	O+

الفصل الثالث: التراكيب الشرطية (Conditional Structures)

نلاحظ ان جميع الامثلة التي مرت بنا سابقاً، نفذت بطريقة متسلسلة(أي خطوة بعد خطوة).وهنا يتبادر الى الذهن السؤال الاتي :

سؤال:- كيف باستطاعتنا نقل تنفيذ خطوات برنامجاً ما بدون تسلسل او بمعنى اخر،كيف يتم التحكم بتنفيذ خطوات البرنامج كيفما نريد؟

للأجابة عن هذا السؤال، علينا دراسة مفهوم التراكيب الشرطية .

(3.1) التركيب الشرطي البسيط (اذا / If)

الصيغة العامة

```
if(condition)
   statement;
```

عمل If :- اذا كان الشرط (Condition) صحيحاً، فنفذ الجملة (Statement).

مثال(3.1.1) :-

Program	التنفيذ
#include<stdio.h> main ( ) { int x; scanf("%d" ,&x) ; if (x > 0) printf(" The number is Positive") ; }	التنفيذ1: **15** The number is Positive التنفيذ2: **-6**

نلاحظ من خلال التنفيذ1 للبرنامج أعلاه، عند إدخال x=15، فان الشرط متحقق وبالتالي ظهرت لنا العبارة

The number is Positive

بينما عند التنفيذ2، أي عند ادخال x= -6 لم تظهر لنا أي عبارة.

مثال(3.1.2) :-

Program	التنفيذ
#include<stdio.h> main ( ) { int x; scanf("%d" ,&x) ; if (x > 0) printf(" The number is Positive") ; if (x < 0) printf("The number is Negative") ; }	التنفيذ1: **15** The number is Positive التنفيذ2: **-6** The number is Negative

نلاحظ من خلال المثال اعلاه، في التنفيذ الاول ظهرت لنا العبارة The number is Positive بينما في التنفيذ الثاني ظهرت لنا العبارة The number is Negative، هنا تم استخدام التركيب الشرطي البسيط مرتين.

ملاحظة(3.1.3):- اذا كان المطلوب تنفيذ أكثر من جملة ضمن شرطاً ما، فيجب وضع هذه الجمل ضمن القوسين { ، } وهذا ما اصطلح على تسميته الجملة المركبة أو البلوك .

```
{
    statement_1;
    statement_2;
    …
    statement_n;
}
```

مثال(3.1.4):- أكتب برنامجاً لقراءة عددين حقيقيين ، ثم رتب هذين العددين تصاعدياً .

Program	التنفيذ
`#include<stdio.h>` `main ( )` `{` `  float  a,b,temp;` `  scanf("%f%f" ,&a ,&b) ;` `  if (a>b)` `  {` `   temp=a ;` `   a=b ;` `   b=temp ;` `  }` `printf( "%f , %f ",a ,b ) ;` `}`	**التنفيذ1:** **5.7 3.94** 3.94 , 5.7 **التنفيذ2:** **-1.6 8.17** -1.6 , 8.17

نلاحظ من خلال المثال أعلاه، عند التنفيذ الاول قد تحقق الشرط وبالتالي تم اجراء التبديل وظهرت النتيجة، بينما في التنفيذ الثاني لم يتحقق الشرط(أي لم يتم أجراء التبديل) وبالرغم من ذلك ظهرت لنا النتيجة صحيحة أيضاً .

(3.2) التركيب الشرطي الكامل (اذا...والا / If...else)

الصيغة العامة

```
if(condition)
   statement_1;
else
   statement_2;
```

عمل If ...else :- اذا كان الشرط (Condition) صحيحاً، فنفذ الجملة (Statement_1) اما اذا كان الشرط غير صحيح فنفذ الجملة(Statement_2).

مثال(3.2.1) :- المطلوب كتابة برنامجاً يتضمن أدخال عدد صحيح، ومعرفة فيما أذا كان هذا العدد زوجي ام فردي.

Program	التنفيذ
`#include<stdio.h>` `main ( )` `{` `int x;` `scanf("%d" ,&x) ;` `if (x%2==0)` `  printf("The number  is Even") ;` `else` `  printf("The number  is Odd") ;` `}`	**التنفيذ1:** **8** The number is Even **التنفيذ2:** **11** The number is Odd

نلاحظ عند التنفيذ1،أي عند ادخال x=8، فان الشرط متحقق وبالتالي ظهرت لنا العبارة The number is Even بينما في التفيذ2،أي عند ادخال x=11، فان الشرط غير متحقق ورغم ذلك ظهرت لنا العبارة The number is Odd.

ملاحظة(3.2.2):- **(1)** التركيب الشرطي الكامل يكافئ تركيبين شرطيين بسيطين .
(2) يمكن استخدام أكثر من تركيب شرطي كامل بصورة متداخلة ضمن البرنامج الواحد ، وهذا ما أصطلح على تسمية التركيب الشرطي المتداخل (Nested Conditional Structure) .

(3.3) تركيب المؤثر الشرطي ? (Conditional Operator ?)

الصيغة العامة

```
(condition)?statement_1:statement_2 ;
```

<u>عمل المؤثر الشرطي (?)</u>:- مشابه إلى عمل التركيب الشرطي الكامل **if…else** .

<u>مثال</u>(3.3.1):- أكتب برنامجاً لحساب وطباعة قيمة Y ، أذا علمت أن

$$Y=\begin{cases} 5-x^2 & \text{if } x \geq 0 \\ 2x^3 & \text{if } x < 0 \end{cases}$$

Program by Conditional Operator?	Program by If…else
#include <stdio.h> main ( ) { float x ,y ; scanf("%d" ,&x) ; y= (x>=0) ? 5-x*x : 2*x*x*x ; printf(" Y =%f ", y) ; }	#include <stdio.h> main ( ) { float x ,y ; scanf("%d" ,&x) ; if(x>=0) y= 5-x*x ; else y= 2*x*x*x ; printf(" Y =%f ", y) ; }

<u>مثال</u>(3.3.2):- أكتب برنامجاً لقراءة عددين صحيحين، ثم يطبع المتغير الذي يحتوي أكبر قيمة .

Program	التنفيذ
#include <stdio.h> main ( ) { int a,b,max ; scanf("%d%d" ,&a ,&b); max= (a>b)? a : b ; printf("The Maximum Number =&d ", max) ; }	**2 7** The Maximum Number = 7

(3.4) التركيب الانتقائي Switch (The selective structure Switch)

كما لاحظنا سابقاً،أن جميع التراكيب الشرطية تتم فيها المقارنة بين قيمتين حيث تكون النتيجة أما صحيحة أو خاطئة، ولكن في بعض الاحيان علينا أن نقارن بين عدد من الحالات تبعاً لشروط مختلفة .

في هذه الحالة نستطيع استخدام التركيب الانتقائي (switch) والذي يقوم باختيار القيمة الصحيحة من عدد من القيم وحسب صحة الشرط الموجود في الصيغة .

الصيغة العامة للتركيب الانتقائي هي :

```
switch (expression)
{
 case value_1:   statements_1 ;
                 break ;
 case value_2:   statements_2 ;
                 break ;
 .
 .
 .
 case value_n:   statements_n ;
                 break ;
 default:        statement_ m ;
}
```

حيث

expression : هو ذلك التعبير الذي يجب أن تكون نتيجته من النوع الصحيح (int) أو من النوع الحرفي(char) .

case : تمثل نوع الحالة المناسبة بعد احتساب التعبير .

value : تمثل قيمة التعبير ويمكن أن تكون عدداً موجباً أو سالباً من النوع الصحيح (int) أو من النوع الحرفي (char) .

break : وهي عبارة توقف، تستعمل عند آخر كل مجموعة جمل من جمل الحالة (case) لتفادي استمرار بقية الحالات(cases)، واذا لم تستعمل بعد أي حالة فان التعبير ينتقل الى الحالة الموالية لهذه الحالة .

default : وتعني حالة اسقاط وهي اختيارية (يمكن عدم ذكرها في البرنامج)، وتنفذ عندما يكون قيمة (expression) لا تتحقق مع أي قيمة (value) .

مثال(3.4.1):- باستخدام تركيب الانتقاء switch ، أكتب برنامجاً لادخال عددين من النوع الحقيقي و مؤثر حسابي يشير الى العملية الحسابية المستخدمة (+ , - , * , /) ، مع طباعة الرسالة المناسبة أذا لم يكن المؤثر المُدخل يشير الى احد العمليات الحسابية الاربعة .

Program	التنفيذ
``#include <stdio.h>```	**التنفيذ 1 :**

Program

```
#include <stdio.h>
main ( )
{
  float a , b ;
  char op ;
  printf(" Enter Two Real Numbers : ") ;
  scanf("%f%f" ,&a ,&b) ;
  printf(" Enter Operator : ")  ;
  scanf("%c",&op) ;
  switch (op)
  {
   case '+' : printf( "%a + %f = %f" ,a,b,a+b ) ;
                break ;
   case '-' : printf( "%a - %f = %f" ,a,b,a-b ) ;
               break
   case'*' :printf( "%a * %f =%f",a,b,a*b );
               break ;
   case '/' : if (b==0)
               printf(" Error Divide by Zero ") ;
               else
               printf( "%a / %f = %f" ,a ,b ,a/b ) ;
               break ;
   Default : printf(" Error Input Operator ") ;
}
```

التنفيذ

التنفيذ 1 :

Enter Two Real Numbers : 2.5   1.7
Enter Operator : *
2.5 * 1.7 = 4.25

التنفيذ 2 :

Enter Two Real Numbers : -0.9   4.11
Enter Operator : +
-0.9 + 4.11 = 3.21

التنفيذ 3 :

Enter Two Real Numbers : 1.5   7.67
Enter Operator : ^
Error Input Operator

التنفيذ 4 :

Enter Two Real Numbers : 2.6   19.2
Enter Operator : /
2.6 / 19.2 = 0.135416

مثال(3.4.2):- أكتب برنامجاً لقراءة متغير صحيح x، ثم أحسب وأطبع قيمة y حيث أن

$$y=\begin{cases} 3x-7 & if \quad x=-3 \\ 5x^2 & if \quad x = 2 \; or \; 5 \\ x-4x^3 & if \; x = \; -4 \; or \; 4 \end{cases}$$

Program

```
#include <stdio.h>
main ( )
{
 int x, y ;
  printf(" Enter Integer Number : ");
  scanf("%d",&x)  ;
  switch (x)
  {
   case -3: y=3*x-7 ;
             printf("\n Y =%f  ", y) ;
             break ;
   case 2 :
   case 5 :  y=5*x*x ;
             printf("\n Y =%f  ", y) ;
             break ;
   case -4 :
   case  4 : y=x-4*x*x*x ;
              printf("\n Y =%f  ", y) ;
              break ;
   Default : printf( "\n Error Data out the Range" );
}
```

التنفيذ

التنفيذ 1 :

Enter Integer Number : 2
Y= 20

التنفيذ 2 :

Enter Integer Number : -6
Error Data out the Range

التنفيذ 3 :

Enter Integer Number : 4
Y= - 132

التنفيذ 4 :

Enter Integer Number : -3
Y= -16

التنفيذ 4 :

Enter Integer Number : 5
Y= 125

نلاحظ في المثال اعلاه، انه تم ادراج الحالات التي لها نفس المعادلة الحالة تلو الاخرى ،والاكتفاء بذكر المعادلة وعبارة الطباعة في نهاية الحالة الاخيرة من الحالات التي لها نفس المعادلة .

تمارين (الفصل الثالث)

تمرين 1 : أكتب برنامجاً يقرأ ثلاث متغيرات من النوع الحقيقي، ثم يطبع هذه المتغيرات بترتيب تصاعدي .

تمرين 2 : أكتب برنامجاً لقراءة أطوال أضلاع مثلث ثم،

a. أطبع كلمة EQULLATERAL في حالة تساوي الأضلاع .
b. أطبع كلمة ISOSCELES في حالة متساوية الساقين .
c. أطبع كلمة SCALENE في حالة اختلاف الأضلاع .
d. أطبع العبارة ERROR TRIANGLE LENGTH في حالة عدم التوافق مع الحالات الثلاث اعلاه .

تمرين 3 : أكتب برنامجاً لادخال متغيرين من النوع الصحيح ومتغير حرفي يدل على الحرف الأول من العملية الحسابية

(Addition, Subtraction, Multiplication, Division) سواء كان الحرف صغيراً أو كبيراً، ثم أحسب وأطبع ناتج العملية الحسابية .

تمرين 4 : المطلوب كتابة برنامج لقراءة المرتب الأساسي لبائع ومقدار مبيعاته، ثم حساب وطباعة الراتب الصافي للبائع، علماً أن الراتب الصافي = الراتب الأساس + المكافئة المئوية . حيث أن المكافئة المئوية تحسب بالصيغة التالية

a. 2% من مرتبة الأساسي، أذا كانت مبيعاته أقل أو تساوي ثلاث أضعاف مرتبة الأساسي.
b. 3% من مرتبة الأساسي، أذا كانت مبيعاته أكثر من ثلاث أضعاف مرتبة الأساسي.
c. 5% من مرتبة الأساسي، أذا زادت مبيعاته على خمسة أضعاف مرتبة الأساسي.

الفصل الرابع: تراكيب التكرار (الحلقات) Iteration Structures (Loops)

تحت شرط معين يتحتم علينا أحياناً تنفيذ جملة أو مجموعة من الجمل عدداً من المرات، وهذا ما يوصف **بالتكرار** . في اللغة C توجد مجموعة من الصيغ التي تتعامل مع مفهوم التكرار، ومن هذه الصيغ هي :   while , do…while , for

(4.1) الحلقة التكرارية بينما (The iteration loop while)

الصيغة العامة	
`while(condition)` `statement ;`	

عمل الحلقة while : أختبر الشرط (condition) أولاً، فأذا كان الشرط صحيحاً، نفذ الجملة (statement)، كرر هذا الاختبار والتنفيذ لغاية أن يصبح الشرط غير صحيح (خاطئاً) .

ملاحظة(4.1.1) :- **(1)** يمكن تكرار تنفيذ أكثر من جملة ضمن الحلقة while ، بشرط أن توضع هذه الجمل ضمن الجملة المركبة (البلوك) .
(2) تستخدم الحلقة while عندما يكون عدد التكرارات معلوم أو غير معلوم .
(3) يمكن استخدام أكثر من حلقة تكرارية بصورة متداخلة ضمن البرنامج الواحد ، وهذا ما أصطلح على تسمية الحلقات التكرارية المتداخلة (Nested Iteration Loop) .

مثال(4.1.2):-

Program	التنفيذ
`/*custom countdown using while*/` `#include <stdio.h>` `main ()` `{` ` int n ;` ` printf( "Enter the starting number : " ) ;` ` scanf("%d",&n) ;` ` while (n>0)` `  {` `   printf( "%d,",n) ;` `   --n;` `  }` ` printf ( "FIRE!" ) ;` `}`	Enter the starting number : 8 8, 7, 6, 5, 4, 3, 2, 1, FIRE!

نلاحظ من خلال تنفيذ البرنامج اعلاه أنه تم تخصيص القيمة 8 للمتغير n ،ثم اختبار الشرط (n>0) ،حيث أن الشرط كان صحيحاً فتم طباعة القيمة 8 ثم أصبحت قيمة n=7 من خلال مؤثر النقصان n-- وبقي الشرط صحيحاً وهكذا تم تكرار العملية الى أن أصبحت قيمة n=0 عندئذ أصبح الشرط غير صحيح وبالتالي تم الخروج من الحلقة نهائياً ،والانتقال الى تنفيذ العبارة FIRE! .

مثال(4.1.3):- أكتب برنامجاً لحساب وطباعة مجموع الأعداد   4.0,4.5,5.0,5.5,…,9.5,10

Program	التنفيذ
`#include <stdio.h>` `main ( )` `{` `   float  a=4.0 , sum=0.0 ;` `  while (a<=10)` `   {` `    sum+=a ;` `    a+=0.5 ;` `   }` `  printf(" The Total =%f ",sum) ;` `}`	The Total = 87.00

مثال(4.1.4):- أكتب برنامجاً لحساب وطباعة المضروب(factorial) للأعداد من 1 الى 10، حيث أن مضروب العدد n(!n) يحسب كالاتي :
n!=n(n-1)(n-2)…3.2.1

Program	التنفيذ
#include <stdio.h> main ( ) { long factorial ; int k ,a=1 ; while (a<=10) { factorial=1 ; k=1 ; while (k<=a) { factorial*=k ; ++k ; } printf("%d!=%d\n ",a ,factorial) ; ++a ; } }	1!= 1 2!= 2 3!= 6 4!= 24 5!= 120 6!= 720 7!= 5040 8!= 40320 9!= 362880 10!= 3628800

في البرنامج اعلاه تم تنفيذ حلقتين تكراريتين، الحلقة ألاولى والتي تسمى بالحلقة الخارجية وهي تنفيذ الجملة المركبة بين القوسين التابعين لها طالما أن العدد a لم يتجاوز 10، تأتي بعدها الحلقة الثانية والتي تسمى الحلقة الداخلية وهي أيجاد مضروب العدد، وذلك من خلال الجملتين

```
factorial*=k ;
++k ;
```

(4.2) الحلقة التكرارية أفعل...بينما (The iteration loop do… while)

```
do                                الصيغة العامة
    statement ;
 while(condition) ;
```

عمل الحلقة do…while : هذه الحلقة مشابه للحلقة التكرارية while وتختلف عنها في أمرين، أولاً أن الحلقة do…while تبدأ التنفيذ للجملة(statement) ثم تتحقق من الشرط بعد ذلك، ثانيا يتم التنفيذ في هذه الحلقة مرة واحدة على الاقل حتى ولو كان الشرط غير متحقق .

مثال(4.2.1):- أكتب برنامجاً لادخال وطباعة عدد صحيح،أوقف البرنامج عند أدخال عند ادخال القيمة 0.

Program	التنفيذ
#include <stdio.h> main ( ) { unsigned long n ; do { printf(" Enter number (0 to end): ") ; scanf("%d%,&n) ; printf(" You entered: %d\n" ,n) ; }while (n != 0) ; }	Enter number (0 to end): 1254 You entered: 1254 Enter number (0 to end): 160277 You entered: 160277 Enter number (0 to end): 33385 You entered: 33385 Enter number (0 to end): 0 You entered: 0

مثال(4.2.2):-

Program	التنفيذ
<pre>#include <stdio.h> main ( ) { int n ; printf("Enter the starting number : ") ; scanf("%d%,&n) ; do { printf("%d , " , n) ; --n; } while (n>0) printf("FIRE!") ; }</pre>	**التنفيذ 1 :** Enter the starting number : 8 8, 7, 6, 5, 4, 3, 2, 1, FIRE! **التنفيذ 2 :** Enter the starting number : 0 0, FIRE!

مثال(4.2.3):- أكتب برنامجاً لحساب وطباعة مجموع الأعداد 4.0,4.5,5.0,5.5,…,9.5,10

Program	التنفيذ
<pre>#include <stdio.h> main ( ) { float a=4.0 , sum=0.0 ; do { sum+=a ; a+=0.5 ; } while (a<=10) ; printf(" The Total =%f " ,sum) ; }</pre>	The Total = 87.00

مثال(4.2.4):- أكتب برنامجاً لحساب وطباعة المضروب(factorial) للأعداد من 1 الى 10 .

Program	التنفيذ
<pre>#include <stdio.h> main ( ) { long factorial ; int k ,a=1 ; do { factorial=1 ; k=1 ; do { factorial*=k ; ++k ; } while (k<=a) printf("%d!=%d\n ",a ,factorial) ; ++a ; } while (a<=10) }</pre>	1!= 1 2!= 2 3!= 6 4!= 24 5!= 120 6!= 720 7!= 5040 8!= 40320 9!= 362880 10!= 3628800

(4.3) الحلقة التكرارية لأجل (The iteration loop for)

الصيغة العامة

```
for(initialization ; condition ; increase)
        statement ;
```

عمل الحلقة for : أولاً نفذ (initialization) والذي يمثل القيمة الأولية لعداد الحلقة(عداد الحلقة هو عبارة عن متغير)، ثانياً أختبر الشرط (condition) ، فأذا كان الشرط صحيحاً، نفذ الجملة (statement)، ثالثاً نفذ (increase) والذي يمثل مقدار الزيادة أو النقصان بمقدار معين في عداد الحلقة، كرر الخطوات ثانياً وثالثاً لغاية أن يصبح الشرط غير صحيح (خاطئاً) .

مثال(4.3.1):-

Program	التنفيذ
`/*countdown using a for loop */` `#include <stdio.h>` `int main ( )` `{` ` int k ;` ` for (k=10; k>0; k--)` `   printf("%d,",k);` `   printf( "FIRE!") ;` `}`	10, 9, 8, 7, 6, 5, 4, 3, 2, 1, FIRE!

مثال(4.3.2):- أكتب برنامجاً لحساب وطباعة المضروب(factorial) للأعداد من 1 الى 10 .

Program	التنفيذ
`#include <stdio.h>` `main ( )` `{` `   long factorial ;` `   int k ,a ;` `   for (a=1;a<=10;a++)` `   {` `      factorial=1 ;` `      for(k=1;k<=a;k++)` `         factorial*=k ;` `      printf("%d!=%d\n ",a ,factorial) ;` `   }` `}`	1!= 1 2!= 2 3!= 6 4!= 24 5!= 120 6!= 720 7!= 5040 8!= 40320 9!= 362880 10!= 3628800

مثال(4.3.3) :- أكتب برنامجاً لحساب وطباعة مجموع الأعداد 4.0,4.5,5.0,5.5,…,9.5,10

Program	التنفيذ
`#include <stdio.h>` `main ( )` `{` `  float a , sum=0.0 ;` `  for( a=4.0;a<=10;a+=0.5)` `  sum+=a ;` `  printf( " The Total = %f", sum ) ;` `}`	The Total = 87.00

مثال(4.3.4):- المطلوب كتابة برنامجاً لحساب وطباعة المعدل لمجموعة من الاعداد الحقيقية والتي عددها n .

Program	التنفيذ
#include <stdio.h> main ( ) { int n, counter ; float number, sum, average ; sum=0.0 ; printf("Enter The Size of The List : ") ; scanf("%d",&number) ; printf("\nThe Data : ") ; for(counter=0;counter<n;counter++) { scanf("%",&number) ; sum+=number ; } average=sum/n ; printf("\n The Average of all Numbers =%f ",average) ; }	Enter The Size of The List : 7 The Data : 12.5 -10.04 35 1.432 7.7 101.1 34.567 The Average of all Numbers = 26.037

(4.4) الجمل التفرعية (Branching Statement)

في بعض البرأمج يتعين علينا تحويل المسار التتابعي لاوامر البرنامج الى الخروج من الحلقات التكرارية أو الرجوع اليها أو الخروج نهائياً من البرنامج، الجمل التفرعية تساعدنا في هذه العملية، ولكن يجب أن يكون أستخدام هذه الجمل محدود،لانه احيانا وفي البرأمج الكبيرة تسبب لنا مشاكل أثناء التنفيذ . من هذه الجمل هي :

(1) جملة أقطع Break : تستخدم هذه الجملة للخروج من الحلقات التكرارية، حيث يتم أنهاء التكرار متى ما وصل التنفيذ الى هذه الجملة. نتذكر بأن هذه الجملة استخدمت لنفس المعنى في موضوع التركيب الانتقائي switch .

مثال(4.4.1) :-

Program	التنفيذ
/*break loop example*/ #include <stdio.h> main ( ) { int n ; for (n=10; n>0; n--) { printf("%d ," ,n) ; if (n==3) { printf("countdown aborted!") ; break ; } } }	10, 9, 8, 7, 6, 5, 4, 3, countdown aborted!

تم التفيذ بطباعة الاعداد من 10 ولغاية 3 ثم طبعت العبارة countdown aborted! بعدها تم التوقف والخروج من الحلقة التكرارية وذلك لكون الشرط قد تحقق في التركيب الشرطي البسيط if .

(2) جملة الاستمرار Continue : تستخدم هذه الجملة بالاستمرار في توجيه التحكم الى نهاية الحلقة وبالتالي الرجوع الى بداية الحلقة وأكمال تنفيذها حتى النهايه .

مثال(4.4.2) :-

Program	التنفيذ
`/* continue loop example*/` `#include <stdio.h>` `main ( )` `{` `  int n ;` `  for (n=10; n>0; n--)` `  {` `   if (n==5)` `   continue ;` `   printf("%d ," ,n) ;` `  }` `  printf( "FIRE!")  ;` `}`	10, 9, 8, 7, 6, 4, 3, 2, 1, FIRE!

(3) جملة خروج Exit : وتعني الخروج نهائياً من البرنامج وترجع بالقيمة صفراً اذا كان البرنامج نفذ على أحسن ما يرام، وبأي قيمة لا تساوي صفراً اذا كان هناك خطأ.

مثال(4.4.3) :-

Program	التنفيذ
`#include <stdio.h>` `main ( )` `{` ` int i, number, sum=0;` ` for (i=1;i<10;i++)` `  {` `   printf("\nEnter value :" ) ;` `   scanf("%d",&number) ;` `   if (number< 0)` `    {` `     printf("This is negative ") ;` `     exit( );` `    }` `    sum+=number ;` `  }` ` printf("The sum of positive numbers %d", sum) ;` `}`	Enter value : 9 Enter value : 4 Enter value : 6 Enter value : 8 Enter value : 1 Enter value : 12 Enter value : -7 This is negative

عندما يكون الشرط الموجود في التركيب الشرطي البسيط if صحيحاً تطبع الرسالة المطلوبة،ثم تنفذ جملة الخروج exit() التي تعني الخروج ليس فقط من الحلقة for بل الخروج نهائياً من البرنامج وبالتالي لن تنفذ الجملة الموالية لهذه الحلقة، وعلية لن يطبع البرنامج مجموع الاعداد الموجبة المدخلة، وهذا ما نشاهدة في التنفيذ اعلاه .

(4) جملة أذهب الى goto : تستخدم هذه الجملة بتحويل المسار التتابعي لاوامر البرنامج الى جملة معينة.

الصيغة العامة **goto lable ;**

حيث label اسم العنوان وينطبق عليه نفس شروط اسم المتغير حيث يمكن وضعه في أي مكان من البرنامج .

مثال(4.4.4) :-

Program	التنفيذ
``/*goto loop example*/```	10, 9, 8, 7, 6, 5, 4, 3, 2, 1, FIRE!

```
/*goto loop example*/
#include <stdio.h>
main ( )
{
 int n=10 ;
 last :
 printf("%d ," ,n) ;
 n--;
 if (n>0)
 goto last ;
 printf( "FIRE!\n" ) ;
}
```

نلاحظ أن الشرط كلما كان صحيحاً فأن العبارة goto متحققة وبالتالي فان التفيذ ينتقل الى العنوان : last الى أن يصبح الشرط غير صحيح عندئذ ينتقل التنفيذ الى طباعة العبارة FIRE! .

تمارين (الفصل الرابع)

تمرين 1 **:** أكتب برنامجاً لحساب وطباعة حاصل جمع مربعات الاعداد الصحيحة الفردية الواقعة بين عددين يتم ادخالهما عن طريق لوحه المفاتيح .

تمرين 2 **:** المطلوب كتابة برنامج لايجاد جميع الاعداد الاولية الواقعة بين *1* و *100* .

تمرين3 **:** أكتب برنامجاً لحساب وطباعة مجموع المتسلسلة التالية :

$$T = \frac{3a}{8!} - \frac{2b}{7!} + \frac{3a}{6!} - \frac{2b}{5!} + \ldots + \frac{3a}{2!} - \frac{2b}{1!}$$

تمرين 4 **:** فصل دراسي به عدد غير معروف من الطلاب، أكتب برنامجاً لقراءة رقم الطالب ID وجنسة *SEX* ودرجاته في ثلاث امتحانات *T1* , *T2* , *T3* ،أوقف تنفيذ البرنامج عند ادخال رقم الطالب قيمة سالبة ، المطلوب :

a. حساب وطباعة معدل الدرجات لكل طالب .
b. حساب وطباعة عدد الطالبات في الصف .
c. أيجاد وطباعة أكبر معدل في الصف .
d. طباعة رقم الطالب ومعدله مع النتيجة المقابلة لمعدله والتي هي على النحو الاتي :

النتيجة	حدود المعدل
Fail	أصغر من 50
Admit table	أكبر من أو يساوي 50 وأصغر من 60
Medial	أكبر من أو يساوي 60 وأصغر من 70
Good	أكبر من أو يساوي 70 وأصغر من 80
Very Good	أكبر من أو يساوي 80 وأصغر من 90
Excellence	أكبر من أ ويساوي 90 وأصغر من 100

الفصل الخامس: المصفوفات (Arrays)

في الفصول التي مرت بنا سابقاً ،كان اسم المتغير يحمل قيمة واحدة فقط قابلة للتغير أثناء تنفيذ البرنامج،في حين أنه هناك مجموعة من البيانات التي لها نفس النوع والتي تحتاج الى عملية تخزين في متغيرات قليلة، ولكي يصبح البرنامج سهل الكتابة والمتابعة والفهم ،علينا استخدام مفهوم المصفوفة (هنا المصفوفة عبارة عن سلسلة من العناصر التي لها نفس النوع والموضوعة في الذاكرة بشكل متجاور والتي تحمل اسماً واحداً بحيث يمكن الرجوع الى هذه العناصر والتعامل معها عن طريق هذا الاسم) .

وتقسم المصفوفات الى قسمين وهما :

القسم ألاول : مصفوفات ذات البعد الواحد (One-Dimensional Arrays) .

القسم الثاني : مصفوفات متعددة الابعاد (Multi-Dimensional Arrays) .

ملاحظة :- في لغة C يوجد الامر #define الذي يعتبر من أوامر المعالج الأولي (Preprocessor) والذي يقوم بأنشاء معرفات تسمى الثوابت الرمزية(Symbolic Constants) حيث يمكن وضعة في أي مكان من البرنامج ولكن من الأفضل ما يوضع في أوله،أي بعد جمل #include . في كثير من البرامج يستخدم هذا الامر في الاعلان عن عدد عناصر المصفوفة .

الصيغة العامة للامر   **#define Constant_name value ;**

مثال :-

```
#define PI 3.141519
#define MAX   15
#define STRING "This is a string"
```

(5.1) المصفوفات ذات البعد الواحد :-

الصيغة العامة للاعلان عن مصفوفة ذات بعد واحد

Type Array_name [index] ;

حيث

Type : يمثل نوع البيانات في المصفوفة.

Array_name : يمثل اسم المصفوفة.

index : يمثل عدد عناصر المصفوفة .

مثال(5.1.1) :- يمكن الاعلان عن 5 قيم صحيحة في مصفوفة ما وبدون استخدام 5 متغيرات مختلفة . فمثلاً   int billy[5] ;
حيث أن اسم المصفوفة هنا billy وعناصرها من النوع الصحيح، نلاحظ الشكل التالي والذي يُظهر كيف يتم حجز مواقع في الذاكرة لعناصر المصفوفة billy .

	0	1	2	3	4
billy					

int

حيث ان كل خانة تضم عنصر من عناصر المصفوفة والذي يكون من النوع الصحيح، دليل(index) هذه القيم رُقم من 0 الى 4 ،لانه دائماً في لغة C العنصر الأول في المصفوفة يكون دليلة 0 .فمثلاً العنصر الاول في المصفوفة هو billy[0] بينما العنصر الثاني هو billy[1] وهذا الى العنصر الاخير في المصفوفة وهو billy[4] . كما نلاحظ في الشكل الاتي :

	billy[0]	billy[1]	billy[2]	billy[3]	billy[4]
billy					

القيم الأبتدائية للمصفوفة ذات البعد الواحد(Initial Values) :- يمكن تخصيص أو شحن قيم مبدئية لأي مصفوفة عن طريق :

أولاً :- اثناء الاعلان عن المصفوفة. والصيغة العامة هي

```
Type Array_name [index]= { value_1,value_2,…,value_n } ;
```

حيث value_1,value_2,…,value_n تمثل قيم المصفوفة على الترتيب. هنا عدد العناصر هو n والذي يمثل index .

مثال(5.1.2) :- لتكن لدينا المصفوفة int billy[5]={ 16, 2, 77, 40, 12071 } ;
هنا billy[0]=16 , billy[1]=2 , billy[2]=77 , billy[3]=40 , billy[4]=12071 ،كما نلاحظ في الشكل الاتي :

	0	1	2	3	4
billy	16	2	77	40	12071

ثانياً :- عن طريق دالة الادخال scanf( ).

مثال(5.1.3) :-

```
for (i=0 ;i<5 ; i++)
  scanf("%d",&billy[i]) ;
```

ثم يتم أدخال القيم عن طريق لوحة المفاتيح .

مثال(5.1.4) :- في البرنامج التالي يتم تخصيص أربعة قيم من النوع الصحيح الى مصفوفة ذات بعد واحد أسمها billy ثم طباعة قيم المصفوفة مع مجموع تلك القيم .

Program	التنفيذ
<pre>/*arrays example*/ #include <stdio.h> #define M 5 main () { int billy [M] = {16, 2, 77, 40, 12071}; int n, result=0; for (n=0 ; n<5 ; n++) { printf(" billy[%d]=%d\n", n, billy[n]) ; result += billy[n]; } printf(" The sum =%d ", result) ; }</pre>	billy[0]= 16 billy[1]= 2 billy[2]= 77 billy[3]= 40 billy[4]= 12071 The sum = 12206

مثال(5.1.5) :- البرنامج التالي يتم فية ادخال وطباعة عناصر مصفوفه ذات بعد واحد وأيجاد وطباعة أصغر عنصر في هذه المصفوفه.

Program	التنفيذ
<pre>#include <stdio.h> #define LEN 5 main ( ) { int i, min ,pos , mat[LEN] ; for(i=0 ; i< LEN ; i++) { scanf("%d",&mat[i]) ; printf(" mat[%d]=%d\n", i, mat[i]) ; } min=mat[0] ; for(i=1 ; i< LEN ; i++) if (mat[i] < min) { min=mat[i] ; pos=i ; } printf(" The Smallest is mat[%d]=",pos, min) ; }</pre>	mat[0]= 3 mat[1]= 8 mat[2]= 7 mat[3]= 2 mat[4]= 4 The Smallest is mat[3]= 2

مثال(5.1.6) :- أكتب برنامجاً لقراءة قيم حقيقية لمصفوفة a عدد عناصرها 6 ، ثم رتب هذه العناصر تصاعدياً وضعها في مصفوفه جديدة b .

Program	التنفيذ
<pre>#include <stdio.h> #define M 6 main ( ) { int i, j ,n1 , n2 ; float temp, a[M] ; for(i=0 ; i< M ; i++) { scanf("%f", a[i]) ; printf(" a[%d]=%f\n", i, a[i]) ; } n1=M-1 ; for(i=1 ; i< n1 ; i++) { n2=i+1 ; for(j=n2 ; j< M ; j++) { if(a[j]-a[i]< 0) { temp=a[i] ; a[i]=a[j] ; a[j]=temp ; } } } printf(" The Sorted Array as following :\n ") ; for(i=0 ; i<M ; i++) printf(" b[%d]=%f\n", i, a[i]) ; }</pre>	a[0]= 2.5 a[1]= 9.1 a[2]= 6.06 a[3]= 2.51 a[4]= 0.05 a[5]= 3.0 The Sorted Array as following : b[0]= 0.05 b[1]= 2.50 b[2]= 2.51 b[3]= 3.00 b[4]= 6.06 b[5]= 9.10

(5.2) المصفوفات متعددة الابعاد :-

سوف نخصص دراستنا في هذا البند حول المصفوفات ذات البعدين (المصفوفات المكونة من صفوف واعمدة) فقط، حيث يمكن أن تعمم الدراسه الى مصفوفات ذات أبعاد أكثر .

الصيغة العامة للاعلان عن مصفوفة ذات بعدين

Type Array_name [index_1][index_2] ;

حيث

Type : يمثل نوع البيانات في المصفوفة .

Array_name : يمثل أسم المصفوفة .

index_1 : يمثل عدد الصفوف في المصفوفة .

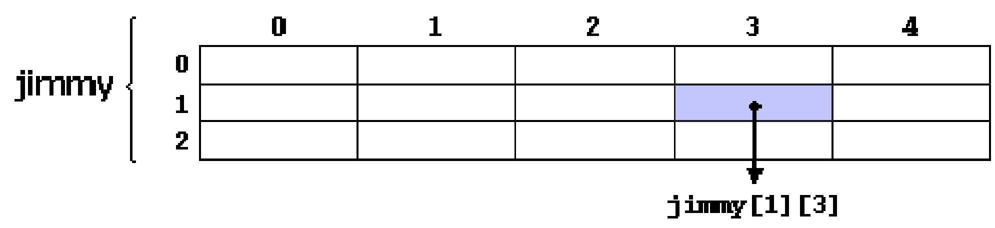
index_2 : يمثل عدد الاعمدة في المصفوفة .

مثال(5.2.1) :- int jimmy [3][5] ;

في هذا المثال، تم الاعلان عن مصفوفة أسمها jimmy والتي تتضمن 3 صفوف و 5 أعمدة (أي أن عدد عناصرها 15 عنصراً).

ملاحظة(5.2.2) :- دليل عناصر المصفوفة(للصفوف والاعمدة)، يُرقم بدأ من 0.

نلاحظ في الشكل التالي مواقع عناصر المصفوفة jimmy في الذاكرة . فمثلاً العنصر jimmy[1][3] هو العنصر الواقع في الصف الثاني العمود الرابع.



<u>القيم الأبتدائية للمصفوفة ذات البعدين</u>(Initial Values) -: يمكن تخصيص أو شحن قيم مبدئية لأي مصفوفة عن طريق :

أولاً -: اثناء الاعلان عن المصفوفة. والصيغة العامة هي

Type Array_name [index_1][index_2]={ { list of first row},{list of second row},…,{list of last row] } ;

<u>مثال</u>(5.2.3) -:

int jimmy[3][5]={ {1,2,3,4,5},{2,4,6,8,10},{3,6,9,12,15} } ;

jimmy

	0	1	2	3	4
0	1	2	3	4	5
1	2	4	6	8	10
2	3	6	9	12	15

ثانياً -: عن طريق دالة الادخال scanf( ).

<u>مثال</u>(5.2.4) -:

Program

```
#include <stdio.h>
#define WIDTH 5
#define HEIGHT 3
main ( )
{
int jimmy [HEIGHT][WIDTH];
int n,m;
for (n=0 ;n<HEIGHT ;n++)
  for (m=0 ;m<WIDTH ;m++)
    {
      scanf("%d",&jimmy[n][m]) ;
    }
for (n=0 ;n<HEIGHT ;n++)
  for (m=0 ;m<WIDTH ;m++)
    {
      printf( "jimmy[%d][%d]=%d\n" ,n ,m ,jimmy[n][m]) ;
    }
}
```

التنفيذ

```
1 2 3 4 5 2 4 6 8 10 3 6 9 12 15

jimmy[0][0]= 1
jimmy[0][1]= 2
jimmy[0][2]= 3
jimmy[0][3]= 4
jimmy[0][4]= 5
jimmy[1][0]= 2
jimmy[1][1]= 4
jimmy[1][2]= 6
jimmy[1][3]= 8
jimmy[1][4]= 10
jimmy[2][0]= 3
jimmy[2][1]= 6
jimmy[2][2]= 9
jimmy[2][3]= 12
jimmy[2][4]= 15
```

نلاحظ أن عناصر المصفوفة في البرنامج السابق ترتبط بعلاقة ما ،لذلك يمكن الحصول على هذه العناصر من خلال تلك والعلاقة وعدم استخدام دالة الادخال cin>> والتي من خلالها قد يحث خطاء في ادخال بعض القيم، لذلك فأن البرنامج السابق يمكن كتابتة بالطريقة الاتية :

مثال(5.2.5) :-

Program	التنفيذ
``#include <iostream.h>``` ```#define WIDTH 5``` ```#define HEIGHT 3``` ```int main ()``` ```{``` ```int jimmy [HEIGHT][WIDTH];``` ```int n,m;``` ```for (n=0 ;n<HEIGHT ;n++)``` ``` for (m=0 ;m<WIDTH ;m++)``` ``` {``` ``` jimmy[n][m]=(n+1)*(m+1);``` ``` }``` ```for (n=0 ;n<HEIGHT ;n++)``` ``` for (m=0 ;m<WIDTH ;m++)``` ``` {``` ``` printf("jimmy[%d][%d]=%d\n" ,n ,m ,jimmy[n][m]) ;``` ``` }``` ```}```	jimmy[0][0]= 1 jimmy[0][1]= 2 jimmy[0][2]= 3 jimmy[0][3]= 4 jimmy[0][4]= 5 jimmy[1][0]= 2 jimmy[1][1]= 4 jimmy[1][2]= 6 jimmy[1][3]= 8 jimmy[1][4]= 10 jimmy[2][0]= 3 jimmy[2][1]= 6 jimmy[2][2]= 9 jimmy[2][3]= 12 jimmy[2][4]= 15

مثال(5.2.6) :- البرنامج التالي يقوم بعملية أدخال مصفوفتين من النوع الصحيح، ثم يقوم باجراء عملية الجمع على المصفوفتين.

$$a = \begin{pmatrix} 2 & -1 & 0 \\ 1 & 3 & 5 \\ 19 & 22 & 8 \end{pmatrix}, b = \begin{pmatrix} 7 & 9 & -3 \\ 8 & -1 & 1 \\ 0 & 0 & 4 \end{pmatrix}$$

Program	التنفيذ
``#include <stdio.h>``` ```#define R 3``` ```#define C 3``` ```main ()``` ```{``` ``` int a[R][C] , b[R][C] ,add[R][C] , mult[R][C] ;``` ``` int i, j , k ;``` ``` for(i=0 ; i< R ; i++)``` ``` for(j=0 ; j< C ; j++)``` ``` scanf("%d%d" ,&a[i][j],&b[i][j]) ;``` ``` for(i=0 ; i< R ; i++)``` ``` for(j=0 ; j< C ; j++)``` ``` add[i][j]= a[i][j]+b[i][j] ;``` ``` printf("Addition of two arrays :\n") ;``` ``` for(i=0 ; i< R ; i++)``` ``` {``` ``` for(j=0 ; j< C ; j++)``` ``` printf("%d \t",add[i][j]) ;``` ``` printf("\n");``` ``` }``` ```}```	2 7 -1 9 0 -3 1 8 3 -1 5 1 19 0 22 0 8 4 Addition of two arrays : 9 8 -3 9 2 6 19 22 12

مثال(5.2.7) :- ليكن لدينا المصفوفة الاتية

$$a = \begin{pmatrix} 2 & -1 & 0 \\ 1 & 3 & 5 \\ 19 & 22 & 8 \end{pmatrix}$$

أكتب برنامجاً لقراءة عناصر المصفوفة a ثم أيجاد وطباعة

(1) حاصل ضرب عناصر القطر الرئيسي للمصفوفة(أي ايجاد a[0][0]*a[1][1]*a[2][2]).

(2) أكبر عنصر في المصفوفة.

Program	التنفيذ
<pre>#include <stdio.h> #define R 3 #define C 3 int main () { int a[R][C]={{2,-1,0},{1,3,5},{19,22,8}} ; int i, j ,max , product ; max=a[0][0] ; product =1 ; for(i=0 ; i< R ; i++) for(j=0 ; j< C ; j++) { if(i==j) product*= a[i][j] ; if(a[i][j] >max) max= a[i][j] ; } printf("The product = %d\n", product) ; printf(" Maximum element in Array =%d ",max) ; }</pre>	The product = 48 Maximum element in Array = 22

نلاحظ من خلال البرنامج اعلاه، انه تم ادخال عناصر المصفوفة بصورة مباشرة،بعدها وضع الشرط الاول التابع للتركيب الشرطي البسيط الاول والمتضمن حساب حاصل ضرب عناصر القطر الرئيسي، بينما الشرط الثاني الموجود في التركيب الشرطي الثاني فيتضمن ايجاد أكبر عنصر في المصفوفة.

(5.3) المصفوفات الحرفية (Character Arrays)

في لغة C يمكن التعامل مع السلاسل الحرفية وكانها مصفوفات يطلق على هذا النوع من المصفوفات بالمصفوفات الحرفية، فمثلاً الكلمة Hello يمكن التعامل معها بشكل مصفوفة حرفية وكما يلي :

```
char myword [6] = { 'H', 'e', 'l', 'l', 'o', '\0' } ;
```

حيث تم الاعلان عن مصفوفة من النوع الحرفي(char) وأسمها (myword) وتتضمن الحروف 'H' , 'e' , 'l' , 'l' , 'o' بالاضافة الى الرمز '\0' والذي يسمى الرمز الصفري، يجب أن يكون هذا الرمز موجود في نهاية أي مصفوفة حرفية، حيث أن الغرض منه هو معرفة مترجم اللغة بنهاية عناصر المصفوفة، وعليه يجب حجز له مكان في الذاكرة، أي أن دليل المصفوفة(index) يضاف له 1 عند الاعلان عن المصفوفة.لاحظ أن الدليل في المصفوفة myword هو 6.

ملاحظة(5.3.1) :- يمكن كتابة المصفوفة الحرفية بصيغة السلسلة الحرفية، ولكن يجب الانتباه الى دليل المصفوفة الذي يجب أن يكون بعدد احرف السلسلة الحرفية مضاف له 1 والذي يكون مخصص للرمز الصفري '\0'. فمثلاً نكتب

```
char myword [6] = "Hello" ;
```

ملاحظة(5.3.2) :- يمكن عدم ذكر دليل المصفوفة اثناء الاعلان عنها. مثلاً نستطيع أن نكتب

```
char myword [] = "Hello" ;
```

مثال(5.3.3) :-

Program	التنفيذ
/*null-terminated sequences of characters*/ #include <stdio.h> main ( ) { char question[] = "Please, enter your first name: "; char greeting[] = "Hello, "; char yourname [80]; printf("%s", question); scanf("%s",&yourname); printf("%s, %s" ,greeting , yourname); }	Please, enter your first name: Mohammed Hello, Mohammed

تمارين (الفصل الخامس)

تمرين 1 : مخزن يتضمن ثلاث اصناف من البضائع،المطلوب كتابة برنامج لادخال رقم الصنف مع عدد مبيعات ذلك الصنف خلال الاشهر الثلاث الاولى من السنة مع حساب وطباعة الاتي :

(1) مجموع مبيعات كل صنف خلال الاشهر الثلاثة.
(2) أكثر الاصناف مبيعاً خلال الاشهر الثلاثة.
(3) مجموع المبيعات لجميع الاصناف خلال الشهر الثاني.

تمرين 2 : أكتب برنامجاً لايجاد عدد مرات تكرار الحرف k في نص معين.

تمرين 3 : المطلوب كتابة برنامج يتضمن ادخال أسم المولود، مكان الولادة، الجنس، تاريخ الولادة(اليوم والشهر والسنة)، ثم عمل الاتي:

i. طبع قائمة بأسماء الذكور مع العناوين بترتيب تصاعدي حسب الاسماء.
ii. طبع قائمة بأسماء الاناث وتاريخ ولادتهن.
iii. طبع قائمة سأسماء الذكور والاناث المولودين قبل 2006/12/31.

تمرين 4 : فصل دراسي يتضمن 50 طالباً وطالبة، المطلوب كتابة برنامجاً لقراءة أسم الطالب ودرجاتة في ثلاث مواد (T1,T2,T3)،جنسه (F أو M) ثم حساب وطباعة مايلي:
a) معدل كل طالب في الدرجات الثلاث.
b) المعدل العام للطلبة.
c) اعلى معدل وطباعتة مع اسم الطالب الذي يمتلك ذلك المعدل.
d) عدد الطالبات الناجحات بالمعدل (درجة النجاح 50 فمافوق).

تمرين 5 : ليكن لدينا المصفوفات الاتية :

$$a=\begin{pmatrix}3 & -1 & -20\\ 1 & 3 & 5\\ 9 & 4 & 8\end{pmatrix},b=\begin{pmatrix}1 & 0 & -3\\ 5 & -1 & 19\\ 7 & 0 & 4\end{pmatrix},c=\begin{pmatrix}1 & 4\\ -3 & 2\\ 3 & 5\end{pmatrix}$$

المطلوب كتابة برنامج لحساب وطباعة :
a) (2a-5b)*c , b*c , a*b .

b) أكبر عنصر في المصفوفات الثلاث مع المصفوفة التي تحوي هذا العنصر.

الفصل السادس: الدوال (Functions)

في السابق كُتبت البرامج على أنها كتلة واحدة، وهذا قد يكون غير مناسب في بعض الاحيان حيث يصعب متابعتها وأصلاحها خصوصاً في البرنامج الطويل.ولذلك يفضل تجزئته الى عدة اجزاء صغيرة كل جزء يؤدي مهمة معينة،ثم تختبر هذه الاجزاء وترتبط مع بعضها لتكون البرنامج الكامل.

لهذا الغرض سوف نستخدم مفهوم الدوال،حيث أن الدالة عبارة عن جملة أو مجموعة من الجمل التي تؤدي وظيفة معينة،ويمكن استدعها من أي نقطة من البرنامج.ومن فوائد هذه الدوال مايلي:

i. تعمل على تجزئة البرنامج الطويل الى أجزاء صغيرة يمكن متابعتها واصلاحها من قبل المبرمج .

ii. تساعد على تلافي عمليات التكرار في خطوات البرنامج التي تتطلب عملاً طويلاًوشاقاً .

iii. توفر الدوال الجاهزة(المخزونة في ذاكرة الحاسب) من مساحة الذكرة المطلوبة .

الصيغة العامة للدالة هي :

```
Type function_name(argument1,argument2 ...)
Types of the parameter list variables ;
{
 types of local variables ;

  function body ;

  return(expression) ;
}
```

حيث

Type : يمثل نوع قيمة الدالة عند رجوعها الى البنامج المنادي .
function_name : يمثل أسم الدالة .
argument1, argument2 ... : تمثل دلائل أو معاملات لاستقبال وارجاع البيانات .
Tapes of the parameter variables : نوع المعاملات المستقبلة والمرجعة للبيانات .
types of local variables : المتغيرات المعلن عنها داخل الدالة وهي محلية (والمتغيرات المحلية هي متغيرات يعلن عنها وتستخدم في حدود الداله ولا يمكن التعرف عليها في البرنامج الرئيسي أو أية دالة أخرى حتى ولو كانت تحمل نفس الاسم).
function body : يمثل جملة أو مجموعة من الجمل .
return : تمثل جملة اعادة قيمة التعبير (expression) حسب نوع الدالة .

ملاحظة(6.1):-

i. يتم الاعلان والتعريف عن الدالة في بداية البرنامج وقبل الدالة الرئيسية main( ) .
ii. يتم استدعاء الدالة داخل البرنامج عن طريق اسمها فقط .
iii. ينبغي الاعلان عن نوع الدالة اذا كانت ترجع بقيمة من النوع غير الصحيح(int)، وفي حالة عدم الاعلان عنها قبل استدعائها تكون قيمتها int تلقائياً . ويستحسن الاعلان عنها في كل الاحوال . مثلا

(1) float larger( )
(2) int temp( )
(3) temp( )

iv. قد يكون للدالة مهمة معينة تؤديها بدون ارجاع قيمة عند انتهائها،أي انها دالة خالية بدون معاملات في هذه الحالة يعلن عنها من النوع الخالي (void). فمثلاً

void printmessage ( )   or   void printmessage (void)

مثال(6.2):- أكتب برنامجاً يستخدم دالة لاجراء عملية الجمع لأي عددين صحيحين .

Program	التنفيذ
/* function example */ #include <stdio.h> int addition (int a, int b) { int r ; r=a+b ; return (r) ; } main ( ) { int z ; z = addition (5,3) ; printf ("The result is %d", z) ; }	The result is 8

نلاحظ اننا قمنا بتعريف دالة للجمع (addition) في بداية البرنامج، والتي لها معاملات صحيحة a , b ثم قمنا بتوضيح عمل الدالة من خلال الجمل الموجودة داخل القوسين { , } ،هنا المتغير الصحيح r يمثل متغير محلي، قمنا بستدعاء الدالة داخل البرنامج عن طريق اسمها فقط.بعدها تم اعطاء القيم 5 للمتغير a و 3 للمتغير b .

```
int addition (int a, int b)
               ↑       ↑
z = addition (  5   ,   3  );
```

بعد ذلك نُفذت جمل الدالة وهي عملية الجمع وظهرت لنا النتيجة وهي القيمة 8 والتي اعطيت للمتغير z وكما يلي :

```
int addition (int a, int b)
↓ 8
z = addition (  5   ,   3  );
```

مثال(6.3):- البرنامج التالي يوضح استخدام دالتين ضمن البرنامج الواحد، احدهما لعملية الضرب والاخرى لعملية القسمة .

Program	التنفيذ
#include <stdio.h> int mul (int a, int b) { return (a*b) ; } float div (float a, float b) { return (a/b) ; } main ( ) { int x=5,y=2 ; float n=5.0,m=2.0 ; printf (" %d*%d=%d\n",x,y, mul(x,y)) ; printf (" %d/%d=%f\n",x,y, div (n,m)) ; }	10 2.5

مثال(6.4):- أكتب برنامجاً يستخدم الدالة الخالية لطباعة العبارة I'm a function! .

Program	التنفيذ
<pre>/*void function example*/ #include <stdio.h> void printmessage ( ) { Printf("I'm a function!"); } main ( ) { printmessage ( ); }</pre>	I'm a function!

مثال(6.5):- أكتب برنامجاً لقراءة أربع متغيرات حقيقية ،ثم استخدم الدالة لايجاد أكبر قيمة من هذه المتغيرات .

Program	التنفيذ
<pre>#include <stdio.h> float large (x,y) float x,y ; { if(x>y) return x ; else return y ; } main( ) { float temp1,temp2,max ; float num1,num2,num3,num4 ; printf("Enter Four Real numbers :") ; scanf("%f%f%f%f", num1,num,num3,num4) ; temp1=large(num1,num2) ; temp2=large(num3,num4) ; max=large(temp1,temp2) ; printf("\nThe Largest number :" ,max) ; }</pre>	Enter Four Real numbers : 5.26 0.044 9.9 0.4 The Largest number : 9.9

مثال(6.6):- أكتب برنامجا لحساب مربع العدد على أن يحسب مربع بدالة مستقلة .

Program	التنفيذ
<pre>#include <stdio.h> int square(int x) { return (x*x) ; } main ( ) { int z ; printf(" Enter value :") ; scanf("%d" ,&z) ; printf ("\n The Square of %d is %d", z, square(z)) ; }</pre>	Enter value : 5 The Square of is 25

مثال(6.7):- أكتب برنامجاً لقراءة ثلاث اعداد من النوع الصحيح، ثم رتب هذه الاعداد تصاعدياً باستخدام الدالة .

Program	التنفيذ
<pre>#include <stdio.h> int swap(x,y) int x, y ; { int temp ; temp=x ; x=y ; y=temp ; } main ( ) { int a, b, c ; printf(" Enter three numbers :") ; scanf("%d%d%d", a , b , c) ; if(a>b) { swap(a,b) ; } if(b>c) { swap(b,c) ; } if(a>c) { swap(a,c) ; } printf("\n Ascending order :%d,%d,%d" ,a ,b, c) ; }</pre>	Enter three numbers : 58 25 99 Ascending order : 25 , 58 , 99

في هذا البرنامج يتم استدعاء دالة التبديل swap(x,y) كلما كان الشرط صحيحاً. عند التفيذ تم استدعاء دالة التبديل swap(58,25) لكون الشرط قد تحقق، لذلك تم التبديل بين القيم 58 و 25، اما في بقية الحالات فقد كانت الشروط غير متحققة وبالتالي لم يتم استدعاء دالة التبديل وعلية ظهرت لنا النتائج كما يلي :

25 , 58 , 99

مثال(6.8) :- أكتب برنامجا لحساب القيمة المطلقة للعدد الحقيقي.

Program	التنفيذ
<pre>#include <stdio.h> float fabs(float x) { if(x>= 0) return (x) ; else return(-x) ; } main ( ) { float a; scanf("%f", a); printf("\n Absolute value of %f is %f" ,a , fabs(a)) ; }</pre>	-19.5 Absolute value of -19.5 is 19.5 255 Absolute value of 255.12 is 255.12 -6698 Absolute value of -6698.0 is -6698.0

(6.9) الدوال الرياضية :- في لغة C توجد مجموعة من الدوال الجاهزة والموجودة ضمن المكتبة القياسية للغة ، ومن هذه الدوال هي، الدوال الرياضية الموجودة ضمن ملف العنوان math.h ، وفيما يلي بعض هذه الدوال :

الدالة	الغرض منها	نوع المتغير	القيمة المرجعة
abs(x)	القيمة المطلقة للعدد الصحيح	int	int
fabs(x)	القيمة المطلقة للعدد الحقيقي	double	double
sqrt(x)	الجذر التربيعي	double	double
pow(x,y)	x بقوة y	double	double
sin(x)	الجيب	double	double
cos(x)	الجيب تمام	double	double
tan(x)	الظل	double	double
asin(x)	معكوس الجيب	double	double
acos(x)	معكوس الجيب تمام	double	double
atan(x)	معكوس الظل	double	double
sinh(x)	الجيب الزائدي	double	double
cosh(x)	الجيب تمام الزائدي	double	double
tanh(x)	الظل الزائدي	double	double
exp(x)	الاسية	double	double
log(x)	اللوغرتيم الطبيعي	double	double
log10(x)	اللوغاريتم العشري	double	double

مثال(6.9.1) :-

Program

```
#include <stdio.h>
#include <math.h>
main ( )
{
 double  a  ;
 printf(" Enter value :") ;
 scanf("%f", a );
 printf("Sqrt  of %d is %d\n" ,a , sqrt(a) ) ;
 printf("%d power 2 is %d\n" ,a , pow(a,2) ) ;
 printf(Absolute value of %d is %d" ,a ,fabs(a)  );
}
```

التنفيذ

```
Enter value : 16
Sqrt of 16 is  4
16 power 2 is 256
Absolute value of 16 is 16
```

مثال(6.9.2):-

Program	التنفيذ
#include <stdio.h> #include <math.h> main ( ) { double x , y ; printf(" Enter value :"); scanf("%f",&x); y= fabs(x)+pow(x,3)-5*sin(3*x-2) ;i printf(" Y=%f " , y); }	Enter value : -3.5 Y= -38.5928

مثال(6.9.3):- أكتب برنامجاً لحساب وطباعة قيمة الدالة f عند العدد الحقيقي x، علماً أن

$$f(x) = \begin{cases} \sqrt{3-\cos 2x} & if\ x \geq 0 \\ x^5 - e^{x^2} & if\ x < 0 \end{cases}$$

Program	التنفيذ
#include <stdio.h> #include <math.h> float f(float x) { If(x>= 0) return(sqrt(3-cos(2*x))) ; else return(pow(x,5)-exp(x*x)) ; } main ( ) { float x ; printf(" Enter value :"); scanf("%f",&x); printf("f(%f)=%f\n",x,f(x)); }	**التفيذ 1 :** Enter real value : 0.7 f(0.7) = 1.4143 **التنفيذ 2 :** Enter real value : -3.2 f(-3.2) = -28694.5654

مثال(6.9.4) :- يوجد جذر للدالة f ضمن الفترة المغلقة [a,b] اذا وفقط اذا كان f(a)*f(b) <0. أكتب برنامجاً لمعرفة فيما أذا كان للدالة

$$f(x) = e^x - x - 1$$

جذراً ضمن فترة معينة ام لا.

Program	التنفيذ
#include <stdio.h> #include <math.h> float f(float x) { return(exp(x)-x-1) ; } main ( ) { float a, b ; scanf("%f%f",&a,&b); if (f(a)*f(b) < 0) printf("There exists Root between%f and %f" ,a , b) ; }	0.5 1.0 There exists Root between 0.5 and 1.0

(6.10) المتغيرات الخارجية(Global Variables) :- تعرفنا سابقاً على المتغيرات المحلية والتي يعلن عنها وتستخدم داخل حدود الدالة ولاعلاقة لها بالدوال الاخرى. أما المتغيرات الخرجية فهي التي تكون معروفه لجميع الدوال الموجودة في البرنامج الرئيسي، حيث يتم الاعلان عنها خارج كل الدوال وقبل الدالة الرئيسية main( ) ولايجوز الاعلان عنها أكثر من مرة.

مثال(6.10.1) :- أكتب برنامجاً لعمل الاتي:

a) ادخال نتيجة الامتحان الاول والثاني لكل طالب في فصل دراسي يضم 5 طلاب.
b) استخدام دالة لحساب أكبر درجة ومعدل وحالة كل طالب على النحو الاتي :
   i. ناجح (PASS) اذا كان معدل الطالب يساوي أو أكبر من 50.
   ii. راسب (FAIL) اذا كان معدل الطالب أقل من 50.

Program

```
#include <stdio.h>
float max , avg ;
char status ;
main( )
{
 int i  ;
 float t1 , t2 ;
 for(i=1 ; i<=5 ; i++)
  {
   printf( " Number of Student : %d\n" , i ) ;
   printf("Enter two numbers :" ) ;
   scanf("%f%f",&t1,&t2);
   calculate(t1,t2) ;
   print_them(t1,t2) ;
  }
   return 0 ;
}
calculate(m1,m2)
float m1,m2
{
  if(m1>m2)
     max=m1;
  else
     max=m2 ;
  avg==(m1+m2)/2 ;
  if(avg >=50)
    status="PASS" ;
  else
    status="FAIL" ;
}
print_them(n1,n2)
float n1,n2 ;
{
   printf(" The result : Maximum=%f\n " ,max);
   printf("Average =%f\n" ,avg) ;
   printf("Status :%s\n" ,status) ;
}
```

التنفيذ

```
Number of Student : 1
Enter two numbers : 72  71
The result : Maximum= 72
Average = 71.5
Status : PASS

Number of Student : 2
Enter two numbers : 45  48
The result : Maximum= 48
Average = 46.5
Status : FAIL

Number of Student : 3
Enter two numbers : 90  86
The result : Maximum= 90
Average = 88.0
Status : PASS

Number of Student : 4
Enter two numbers : 50  51
The result : Maximum= 51
Average = 50.5
Status : PASS

Number of Student : 5
Enter two numbers :  25.5   22.5
The result : Maximum= 25.5
Average = 24.0
Status : FAIL
```

في هذا البرنامج تم الاعلان عن المتغيرات الخارجية avg , max من النوع الحقيقي و status من النوع الحرفي وذلك قبل الدالة الرئيسية main( ) وبالتالي جميع هذه المتغيرات شاملة ومعروفة في البرنامج وبقية الدوال المستخدمة من قبل هذا البرنامج.

البرنامج كما نلاحظ قام باستدعاء دالتين، الدالة الاولى calculate(m1,m2) لحساب الدرجة الاكبر والمعدل مع الحالة، بينما الدالة الثانية print_them(n1,n2) فقد استخدمت لطباعة النتائج فقط .

(6.11) **استدعاء الدالة لنفسها أو لدالة أخرى** -: في لغة C يمكن استدعاء الدالة لنفسها أو لدالة أخرى في نفس البرنامج .

مثال(6.11.1) -: أكتب برنامجاً يستخدم دالة لحساب مضروب العدد (factorial) .

Program	التنفيذ
<pre>#include <stdio.h> long factorial (long a) { if (a > 1) return (a * factorial (a-1)) ; else return (1) ; } main ( ) { long number ; printf("Please type a number:\n ") ; scanf("%d",&number) ; printf("%d! =%d " ,number ,factorial (number)); }</pre>	**التنفيذ 1 :** Please type a number: 4 4! = 24 **التنفيذ 2 :** Please type a number: 9 9! = 362880

في هذا البرنامج تم استدعاء الدالة لنفسها، فلاحظ المطلوب حساب factorial(a) لذلك عند الاستدعاء اصبح لدينا استدعاء اخر هو factorial(a-1) ، فمثلاً

factorial(4) = 4*factorial(3)
= 4*3*factorial(2)
= 4*3*2*factorial(1)
= 4*3*2*1
= 24

مثال(6.11.2) -: أكتب برنامجاً لحساب وطباعة المضاعف المشترك الاصغر(least common multiple) لعددين صحيحين .

المضاعف المشترك الاصغر(LCM) يحسب من القانون الاتي :

$$LCM(a,b) = \frac{a*b}{GCD}$$

GCD يمثل القاسم المشترك الاكبر للعدين a و b .

هناك عدة طرق يمكن من خلالها الحصول على القاسم المشترك الاكبر GCD ومنها هذه الطريقة البسيطة :

i. نقوم بتحليل كل عدد الى قواسمه و ثم نبحث عن القواسم التي اشترك فيها العددان .

ii. نضرب القواسم المشتركة فينتج (القاسم المشترك الاكبر) .

فمثلاً :

العدد 24=2*2*2*3

والعدد 28=2*2*7

ومن هنا اشترك العددان في القاسم 2 مرتين وبالتالي

2*2=4 هو القاسم المشترك الاكبر بين العددين 28 , 24.

Program	التنفيذ
(code below)	Enter two numbers : 91 169 LCM = 1183 GCD = 15

```
#include <stdio.h>
least_common_multiple(a,b)
int a,b ;
{
  int c,d ;
  c=a ;
  d=b ;
  return( c*d/greatest_common_divisar(a,b) ) ;
}
greast_common_divisor(x,y)
int x,y ;
{
  int temp ;
  if(x==y)
    return x ;
  else
     if(y>x)
     {
       temp=y ;
       y=x ;
       x=temp ;
     }
  return( greatest_common_divisor(x-y,y)) ;
}
main ( )
{
  int m,n,l,g ;
  printf( " Enter two numbers : " );
  scanf("%d%d",&n,&m);

  l=least_common_multiple(m,n) ;
  g=greatest_common_divisor(m,n) ;
  printf( " LCM =%d\n ", l) ;
  printf(" GCD =%d\n" , g) ;
}
```

خلال البرنامج اعلاه، تم تعريف دالة المضاعف المشترك الاصغر للعددين الصحيحين a,b ، والتي خلالها تم استدعاء دالة القاسم المشترك الاكبر للعدين الصحيحين a,b والتي تم تعريفها كدالة ثانية في البرنامج. بعدها تم ادخال قيم للمتغيرات m,n خلال البرنامج حيث اعطيت هذه القيم الى المتغيرات a,b لحساب القاسم المشترك الاكبر والمضاعف المشترك الاصغر.

(6.12) الدوال والمصفوفات (Functions and Arrays) :-

ذكرنا سابقاً أنه عند معالجة بيانات كثيرة جداً يتحتم علينا استخدام المصفوفات ليسهل معها عملية تخزين البيانات في متغيرات قليلة، ولكي يصبح البرنامج سهل الكتابة والمتابعة والفهم يتم استخدام الدوال مع المصفوفات.

ولكن يجب أن نأخذ في الاعتبار كيفية تمرير عناصر المصفوفة الى أي دالة، وهذا قد يحدث كالاتي :

i. يمكن أن تتم عملية تمرير البيانات الى الدوال بأستخدام المتغيرات الخارجية.

ii. قد يحدث التمرير بالاعلان عن المصفوفة في الدالة بحيث تكون من نفس النوع والطول.

مثال(6.12.1) :- أكتب برنامجاً لقراءة قيم صحيحة وتخزينها في مصفوفة a ذات بعد واحد ثم تخزين هذه البيانات بالعكس في مصفوفة b عن طريق الدالة.

Program	التنفيذ
``#include <stdio.h>``` ... (see code below)	(see output below)

```
#include <stdio.h>
#define M  5
int i , k, a[M] , b[M] ;
int main ( )
{
 for(i=0 ; i< M ; i++)
 {
   printf (" Enter a[%d]= ",i) ;
   scanf ("%d",&a[i]) ;
 }
 reversed_a( ) ;
 printf ("Array after reversed is : \n" ;
 for(i=0 ; i< M ; i++)
 printf (" b[%d]=%d",i,b[i]) ;
 printf ("\n") ;
 return 0 ;
}
reversed_a( )
{
 k=M-1 ;
 for(i=0 ; i< M ; i++)
   {
     b[i]=a[k] ;
     k-=1 ;
   }
}
```

```
Enter a[0]= 7
Enter a[1]= -3
Enter a[2]= 8
Enter a[3]= 4
Enter a[4]= 2
Array after reversed is :
b[0]= 2
b[1]= 4
b[2]= 8
b[3]= -3
b[4]= 7
```

تمارين (الفصل السادس)

تمرين 1 : أكتب برنامجاً لقراءة اطوال مثلث A,B,C ثم أحسب مساحة هذا المثلث عن طريق الدالة، علماً أن

$$AREA = \sqrt{S(S-A)(S-B)(S-C)}$$

$$S = \frac{(A+B+C)}{2}$$

تمرين 2 : أكتب برنامجاً لدالة مهمتها ايجاد وطباعة جميع الاعداد الاولية الواقعة بين *20* و *500* .

تمرين 3 : سلسلة فيبوناشي *Fibonacci Series* وهي عبارة عن سلسلة أرقام يكون فيها الرقمين الاول والثاني يساوي 1 والارقام الثالث ومافوق كل رقم منها يساوي مجموع الرقمين السابقين له .

1 1 2 3 5 8 13 21 33 …

أكتب برنامجاً باستخدام الدالة لحساب مجموع متسلسلة فيبوناشي

تمرين 4 : المطلوب كتابة برنامجاً لحساب معدل N من القيم الصحيحة على أن يتم حساب المعدل في دالة مستقلة.

تمرير 5 : أكتب برنامجاً لقراءة المتغيرين *n,m* ثم حساب قيمة المعادلة

$$p = \frac{n!}{(n-m)!}$$

-

تمرين 6 : أكتب برنامجاً لادخال رقم الموظف وأسمة و راتبة الشهري وعدد ساعات العمل الاضافي التي عملها، ثم حساب الراتب الصافي لعدد N من الموظفين في مؤسسة ما، علماً أن الراتب الصافي للموظف = الراتب الشهري + مبلغ العمل الاضافي . مبلغ العمل الاضافي يحسب عن طريق الجدول الاتي :

ساعات العمل خلال الشهر	المبلغ في الساعة الواحدة(بالدينار)
أقل أو يساوي 10 ساعات	1000
أكثر من 10 ساعات	1500
أكثر من 25 ساعة	2000

على أن تكون النتائج بالشكل الاتي :

رقم الموظف	أسم الموظف	الساعات الاضافية	أجرة الساعة الواحدة	الراتب الصافي
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-

تمرين 7 :- أكتب برنامجاً لقراءة المصفوفة *A(25,25)* التي تحتوي على قيم حقيقية،ثم كتابة دالة لعمل كل مما ياتي :

a) جمع العاصر الواقعة أعلى القطر الرئيسي من المصفوفة.

b) جمع العناصر الواقعة أسفل القطر الرئيسي من المصفوفة.

c) حاصل ضرب عناصر القطر الرئيسي من المصفوفة.

d) ايجاد أكبر عنصر في المصفوفة.

e) ايجاد أصغر عنصر في المصفوفة.

f) ترتيب عناصر المصفوفة بصورة تصاعديا.